Design of Small Rate, Close to Ideal, GLDPC-Staircase AL-FEC Codes for the Erasure Channel

Ferdaouss Mattoussi, Vincent Roca, Bessem Sayadi

Design of Small Rate, Close to Ideal, GLDPC-Staircase AL-FEC Codes for the Erasure Channel

Ferdaouss Mattoussi* Vincent Roca* Bessem Sayadi‡
*Inria, France ‡Alcatel-Lucent Bell Labs, France
{ferdaouss.mattoussi,vincent.roca}@inria.fr, bessem.sayadi@alcatel-lucent.com

***Abstract*—This work introduces the Generalized Low Density Parity Check (GLDPC)-Staircase codes for the erasure channel, that are constructed by extending LDPC-Staircase codes through Reed Solomon (RS) codes based on "quasi" Hankel matrices. This construction has several key benefits: in addition to the LDPC-Staircase repair symbols, it adds extra-repair symbols that can be produced on demand and in large quantities, which provides small rate capabilities. Additionally, with selecting the best internal parameters of GLDPC graph and under hybrid Iterative/Reed-Solomon/Maximum Likelihood decoding, the GLDPC-Staircase codes feature a very small decoding overhead and a low error floor. These excellent erasure capabilities, close to that of ideal, MDS codes, are obtained both with large and very small objects, whereas, as a matter of comparison, LDPC codes are known to be asymptotically good. Therefore, these properties make GLDPC-Staircase codes an excellent AL-FEC solution for many situations that require erasure protection such as media streaming.**

I. Introduction

Low Density Parity Check (LDPC) codes have been intensively studied during the last decade due to their near-Shannon limit performance under iterative Belief-Propagation (BP) decoding [1–3]. A (N, K) LDPC code, where N is the code length and K is its dimension, can be graphically represented as a bipartite graph with N "variable nodes" (VNs) and $M = N - K$ "check nodes" (CNs). Equivalently, LDPC codes can be represented through their H_L parity check matrix translating the connection between (VNs) and (CNs). The degree of a VN or a CN is defined as the number of edges connected to it. A VN of degree n can be interpreted as a "Length Repetition Code" $(n, 1)$, i.e. as a linear block code repeating n times its single information bit towards the CN set. Similarly, a CN of degree n can be interpreted as a Single Parity Check (SPC) code $(n, n-1)$, i.e. as a linear block code associated with one parity equation.

To improve error floor, minimal distance and decoding complexity performances, a generalization of these codes was suggested by Tanner in [3], for which subsets of the variable nodes obey a more complex constraint than an SPC constraint. The SPC check nodes in a GLDPC structure are replaced with a generic linear block codes (n, k) referred to as sub-codes or component codes while the sparse graph representation is kept unchanged. More powerful decoders at the check nodes have been investigated by several researchers in recent years after the work of Boutros et al. [4] and Lentmaier and Zigangirov [5] where BCH codes and Hamming codes were proposed as component codes respectively. Later several works, on several channels, have been carried out in order to afford very large minimum distance and exhibit performance approaching Shannon's limit. Each construction differs to other by modifying the linear block codes (components codes) on the check nodes such as [6–11], or/and the distribution of the structure of GLDPC codes [6] to offer a good compromise between waterfall performance and error floor under iterative decoding.

Recently, a construction of GLDPC using LDPC Staircase code as base code and Reed-Solomon (RS) codes as component codes has been proposed in [12]. This construction allows each component code to produce a potentially large number of repair symbols in terms of RS codes (named *extra-repair symbols*) on demand, a feature that is well suited to situations where the channel conditions can be worse than expected, or to fountain like content distribution applications. The production of these extra-repair symbols allow to extend the initial LDPC Staircase code (based code) to a GLDPC code and *very small rates* are easily achievable. In this work the performance of the GLDPC-Staircase codes are assessed through the use of an Iterative IT/RS decoding. However it is shown that the extra-repair symbols don't totally solve the problem of small stopping sets that stuck the IT decoding process (i.e. IT remains sub-optimal).

In this paper, we revisit the [12] proposal by improving the way of generating the extra-repair symbols and also by analyzing its performance in a finite length behavior. First, the extra-repair symbols are now generated through the use of "quasi" Hankel Matrix. The use of "quasi" Hankel matrices is well adapted to situations where the RS code parameters are dynamic determined. Thanks to the "quasi" Hankel matrix structure, the generator matrix creation complexity is significantly reduced without compromising its MDS characteristics. Second, the decoding algorithm is improved through the use of Maximum Likelihood (ML) decoder. In order to reduce the decoding complexity, we introduce an IT/RS/ML decoding scheme, called here hybrid decoding. Through it, the configuration parameters of the GLDPC codes such as base code rate, source variable nodes degree, and the extra-repair symbols

distribution are discussed.

Simulation results show that our GLDPC construction features excellent decoding performance (i.e. good waterfall region and small error floor) and approach channel capacity even for small objects, both in terms of average decoding overhead and error floor. Additionally, the proposed GLDPC codes can easily be tuned to behave either like predefined rate LDPC-Staircase codes at one extreme, or like a single Reed Solomon code at another extremism, or like small rate codes. This flexibility makes it possible to tune GLDPC-Staircase codes to better match each use-case requirements, making GLDPC-Staircase codes an almost universal Application-Level FEC (AL-FEC) solution.

The paper is organized as follows. Following the introductory section, section II focuses on the design of GLDPC Staircase codes based on RS codes using "quasi" Hankel matrices. Section III provides a performances evaluation of these codes. Finally, we conclude.

II. Proposed low rate coding schema

In this section we first introduce the GLDPC-Staircase code design, as well as their encoding and decoding.

A. GLDPC-Staircase code construction

The GLDPC-Staircase (N_G, K) codes studied in this paper can be represented by a Tanner graph (Figure 1) with the following meaning:

- each **check node** corresponds to a RS code based on "quasi" Hankel matrix, a specific construction of RS codes that has the interesting property that the first repair symbol is **also** equal to the XOR sum of the source symbols. This symbol can therefore be encoded either by means of an LDPC-Staircase encoding (faster) or RS encoding. This property does not hold for the other repair symbols, called extra repair symbols;
- the **variable nodes** are broken into three categories: (1) the source symbols; (2) the first repair symbol generated by each RS code (or by the LDPC-Staircase code), that only depend on source and repair symbols (i.e. each repair symbol depends on the previous repair symbol because of the staircase structure of the LDPC code); and (3) the extra-repair symbols generated by RS codes.

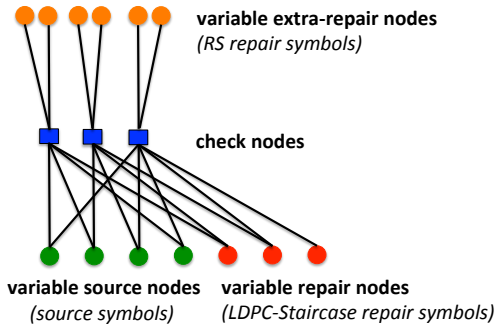


Fig. 1. GLDPC-Staircase $(13, 4)$ code with two extra repair symbols per check node ($E = 2$).

For the reasons detailed in section III-B1 (i.e. improved performance under ML decoding), we assume that all the check nodes have the same number, E, of extra-repair symbols. So for a fixed LDPC-Staircase base code rate, r_L, the code rate of the GLDPC-Staircase code is given by:

$$r_G = \frac{r_L}{1 + (1 - r_L)E} \quad (1)$$

Let N_L and K be the LDPC-Staircase code length and dimension, and N_G be the length of the GLDPC-Staircase code (which is also of dimension K). Then $N_G = N_L + n_{\text{extra}}$, where n_{extra} is the total number of extra-repair RS symbols.

Let H_L be the binary parity-check matrix of the LDPC-Staircase code, of size $M_L = N_L - K$ rows and N_L columns. H_L has the form $(H_1|H_2)$. H_1 is the $M_L \times K$ left-hand side part (information part) and each column is of degree N_1 (number of "1s" per column). H_2 is the $M_L \times M_L$ right-hand side part (redundancy part) and features a staircase (i.e. double diagonal) structure. H_1 is created in a fully regular way, in order to have constant column and row degrees. More precisely, each column of H_1 is of degree N_1 (number of "1s" per column), which is an input parameter during the LDPC-Staircase code creation [13]. Each row m of H_1 is of degree $\frac{N_1}{\frac{1}{r_L}-1}$, and because of the staircase structure of H_2, depending on whether $m = 1$ or $m > 1$, a row m of H_L is of degree $d_1 = \frac{N_1}{\frac{1}{r_L}-1} + 1$ and $d_{m>1} = \frac{N_1}{\frac{1}{r_L}-1} + 2$.

The E extra-repair symbols associated to the m^{th} row of H_L are generated by RS(n_m, k_m) encoding over $GF(2^8)$. Here n_m and k_m are respectively the RS code length and dimension, and they are related to the other parameters of row m as follows. For row $m > 1$, the various source symbols (i.e. from the user point of view) that are involved in this row plus the previous repair symbol are considered as source symbols from the RS point of view. The new LDPC-Staircase repair symbol for this row plus the E extra-repair symbols are considered as repair symbols from the RS point of view. For the first row the only difference is the fact there is no previous repair symbol (it's the beginning of the staircase). So: $n_m = k_m + 1 + E$, with $k_m = d_m - 1$ (no matter the row).

B. Encoding algorithm of GLDPC codes

Two types of repair symbols are produced during encoding: M_L LDPC-Staircase repair symbols, $(p_1, \cdots, p_{M_L})$, and n_e extra-repair symbols, $(e_1, \cdots, e_{n_e})$.

Let $S = (s_1, s_2, \cdots, s_K)$ be the source symbols. The $(p_1, \cdots, p_{M_L})$ repair symbols are computed us usual, by "following the stairs" of the H_L matrix. Moreover, for each row m in H_L, p_m is the XOR sum of LDPC source symbols $x = (x_1, \cdots, x_{K_m})$, where x is a subset of S that correspond to a 1 coefficient in row m of H_L, (and the LDPC repair symbol p_{m-1} if $m \neq 1$).

For each row m, the E extra-repair symbols are computed by multiplying the k_m symbols of LDPC symbols by the systematic generator matrix G_m of an RS (n_m, k_m) code based on an "quasi" Hankel matrix. As mentioned in the previous

section, the k_m symbols are defined by x symbols plus p_{m-1} (if $m \neq 1$). The construction of this systematic matrix is simply obtained by concatenating the identity matrix I_{k_m} with the $k_m \times (n_m - k_m)$ matrix A_{k_m,n_m-k_m}, where A_{k_m,n_m-k_m} is a rectangular sub-matrix of the maximal triangular array "quasi" Hankel matrix. For more details on RS codes based on "quasi" Hankel matrices, please refer to [14][15]. For GLDPC-Staircase codes, this choice has the advantage that p_m can be considered indifferently as an LDPC-Staircase or RS repair symbols [1].

An advantage of this encoding approach is the fact that extra-repair symbols can be produced incrementally, on demand, rather than all at once (unlike LDPC-Staircase repair symbols). Their number can also be rather high since it is only limited by the finite field size, usually GF(2^8). Said differently, GLDPC-Staircase codes can easily and dynamically be turned into small rate codes.

C. Decoding algorithm of GLDPC codes

The decoding of GLDPC codes, named hybrid decoding, consists of a joint use of a four decoders

- *IT decoder over the binary LDPC-Staircase system:* extra-repair symbols are ignored at this step. This solution features a linear complexity with sub-optimal erasure recovery capabilities;
- *RS decoder for a given check node:* this is a classic RS decoding that takes into account the three types of symbols. It has a higher complexity but is MDS;
- *Binary ML decoder over the LDPC-Staircase system:* extra-repair symbols are once again ignored at this step. If this solution features a quadratic complexity in terms of the number of `XOR` operations between symbols, it allows to reach the maximum correction capabilities when ignoring extra-repair symbols;
- *Non binary ML decoder:* this solution also features a quadratic complexity but operations are now significantly more complex (performed on $GF(2^8)$) than simply `XORing` two symbols. However it allows reaching the maximum correction capabilities of the code. It is equivalent to the ML decoding over the full system mentioned above, but the system on which it is applied is hopefully simplified by the previous three decoders.

Decoding succeeds if one or several of these decoders succeed and recover all the missing source symbols.

This description is detailed in ***Algorithm 1*** where $N_b \in \mathbb{N}$ source or/and LDPC-Staircase or/and extra-repair symbols are received.

Algorithm 1 Hybrid decoding of (N_G, K) GLDPC-Staircase codes

for $s = 1$ to N_b **do**
  Step1: Decoding_with_new_symbol(s)
  if (s is LDPC symbol) **then**
    $Select\_IT\_decoding\_with\_received\_symbol()$
    if $IT\_decoding\_successful$ **then**
      $Recover\_symbol$
      $Go\_to\ Step1(s \leftarrow recovered\_symbol)$
    else
      $Select\_RS\_decoding\_with\_received\_symbol()$
      if $RS\_decoding\_successful$ **then**
        $Recover\_symbols$
        $Go\_to\_Step1\ (s \leftarrow each\_recovered\_symbol)$
      end if
    end if
  else if (s is extra repair symbol) **then**
    $Select\_RS\_decoding\_with\_received\_symbol()$
    if $RS\_decoding\_successful$ **then**
      $Recover\_symbols$
      $Go\_to\_Step1\ (s \leftarrow each\_recovered\_symbol)$
    end if
  end if
  END_Decoding_with_new_symbol(s)
end for
if (All or some of source symbols are still erased) **then**
  Step2: ML_decoding
  $Select\_Binary\_ML\_decoding$
  if (All or some of source symbols are still erased) **then**
    $Select\_Non\_Binary\_ML\_decoding$
    if (All or some of source symbols are still erased) **then**
      return $Decoding\_is\_finished\_with\_failure$
    else
      return $Decoding\_is\_completed\_successfully$
    end if
  else
    return $Decoding\_is\_completed\_successfully$
  end if
else
  return $Decoding\_is\_completed\_successfully$
end if

III. Performance evaluation results

A. Experimental setup

We have developed a GLDPC-Staircase codec based on $RS(2^8)$ codes ("quasi" Hankel matrices) under hybrid decoding, in C language, using the `OpenFEC.org` project (`htpp://openfec.org`).

[1]Technically speaking, this is made possible by the fact that the coefficients on the first column of G_i are all equal to 1.

Experiments are carried out considering a memoryless channel along with a transmission scheme where all the source and repair symbols are sent in a fully random order. This choice has the benefit to make the performance results independent of the loss model. Instead the target channel loss rate is the only parameter that needs to be considered.

We determined the performances of these codes by testing several LDPC-Staircase matrix that are generated randomly. For each test, a different LDPC-Staircase matrix is used (more precisely we change the PRNG seed used to create the matrix) and then the results, averaged over tests, show the average behavior of GLDPC-Staircase codes.

Performance evaluations make use of the following two metrics:

- Decoding inefficiency ratio and decoding overhead: the decoding inefficiency ratio is defined as the ratio between the number of symbols needed for a successful decoding over the number of source symbols : ineff $= \frac{nb_{symbols\_needed}}{K} = 1 + \varepsilon$ where ε is the transmission

overhead (ε is often expressed as a percentage). An MDS code, when there is a single block, has a decoding inefficiency ratio equal to 1 (i.e. any K subset of symbols are sufficient for decoding to succeed). With non-MDS codes (e.g. LDPC-Staircase codes) we are usually considering average values, and on average decoding is successful after receiving $K(1+\varepsilon)$ symbols (values for a target decoding success, say 99%, can also be considered instead of the average 50% target we use).

- Decoding failure probability: decoding is said to fail when one or more erased source symbols are not recovered after having received all the non-erased encoding symbols and having launched the decoder. This metric defines at a receiver as a function of the number of received symbols, or equivalently the percentage of symbols lost over the network.

Since the hybrid decoding scheme we introduced is a Maximum Likelihood (ML) decoder (i.e. provide the same erasure recovery performance), we only mention "ML decoding" in the remaining of this paper. The hybrid approach will only impact decoding complexity and speed, which is not addressed in this work.

B. Preliminary tests and internal parameters set up

GLDPC-Staircase codes depend on three important internal parameters namely the extra-repair symbols distribution across the H_L rows, the base code rate r_L, and the N1 parameter of the base code. We begin by analyzing the impact of these three parameters on the erasure recovery performance of GLDPC codes, then using the best values of these parameters we evaluate the performance gains made possibly by the use of the decoding scheme, ML versus IT/RS.

1) Performance as a function of the extra-repair symbols distribution: The distribution of extra-repair symbols per check node is an important parameter. In [12] it is shown that GLDPC codes with IT/RS decoding perform the best under a uniform (irregular) distribution rule. However in our work we consider an ML decoding scheme and the situation is completely different. Therefore we tested the uniform (irregular) distribution of [12] against a (regular) constant distribution, where the same number of extra-repair symbols are generated for each parity check matrix row. Figure 2 shows the average decoding inefficiency ratio (i.e, average of 1000 GLDPC codes with r_G=$\frac{1}{2}$) for the proposed GLDPC codes, for different object sizes. It shows that the constant distribution performs significantly better under ML decoding, both with small and large objects. Therefore in the remaining of this work we only focus on this constant distribution, where there are exactly $e_m = E$ extra-repair symbols per row.

2) Performance as a function of the base code rate r_L: Let us consider a GLDPC-Staircase rate r_G. Several values of the base code rate r_L, or equivalently of E, enable to achieve r_G (see Eq. 1). However choosing a value impacts the performance achieved. In Figure 3, we plot the average decoding inefficiency ratio (i.e, average of 1000 GLDPC codes with r_G=$\frac{1}{3}$) for different object size. This figure provides

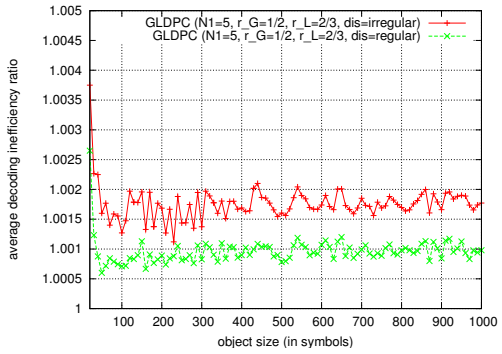


Fig. 2. Performance for uniform (irregular) versus constant (regular) distributions of extra-repair symbols, with N1= 5, $r_L = \frac{2}{3}$ and hybrid decoding

r_G	**E = 0**	**E= 1**	**E= 2**	**E= 3**	**E= 4**	$\delta_{\mathbf{sh}}$
1/3	0.6634	0.6652	0.6664	0.6665	0.6666	0.6667
1/2	0.4946	0.4993	0.4999	0.4999	0.4999	0.5000
2/3	0.3301	0.3330	0.3333	0.3333	0.3333	0.3333

TABLE I
$\bar{\delta}^{ML}$ OF GLDPC CODES ($r_G = \{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$) AS A FUNCTION OF r_L

that increasing rate r_L (i.e, increasing the number E), the average decoding inefficiency ratio quickly approaches 1 (i.e. no overhead, decoding is possible with exactly K symbols) as $E = 3$ (i.e, $r_L = \frac{2}{3}$), even for very small code dimensions. Based on EXtrinsic Information Transfer (EXIT) functions, asymptotic analysis technique, the impact of this parameter on GLDPC codes performances using the upper bound on the ML decoding threshold $\bar{\delta}^{ML}$ is studied. For more details on the asymptotic analysis of GLDPC-Staircase codes please refer to [16]. Table I provides, for each r_G, as seen in Figure 3 that increasing the base code rate r_L (i.e, increasing E) quickly increases the upper bound on the ML threshold until it reaches a stable value (i.e. very close or equal to the Shannon limit δ_{sh}). For $r_L = \frac{2}{3}$ (i.e, $E = 1$ for $r_G = \frac{1}{2}$, and $E = 3$ for $r_G = \frac{1}{3}$) the $\bar{\delta}^{ML}$ very close to δ_{sh}, and above this base code rate $\frac{2}{3}$ the upper bound on the ML threshold improves slightly.

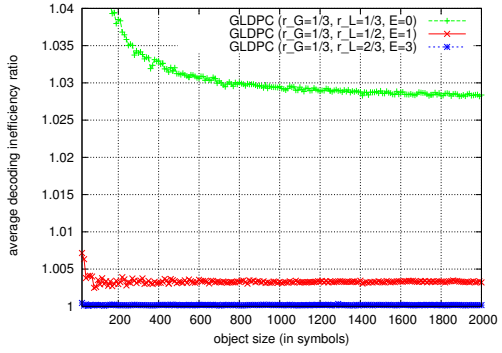


Fig. 3. Performance as a function of base code rate r_L with N1= 5 and hybrid decoding.

During the following tests we chose to set the base code rate equal to $r_L = 2/3$.

3) Performance versus N1: Let us now adjust the third internal parameter of the GLDPC-Staircase codes, N1, which represents the source variable nodes degree of the base code matrix H_L.

Table II provides the average decoding inefficiency ratio (i.e, average of 1000 GLDPC codes for each r_G) for different values of N1 under IT/RS and ML decoding of GLDPC codes with $K = 1000$. Table II shows that, under IT/RS decoding, increasing N1 induces an increase in the average decoding inefficiency ratio because the increase of N1 causes the "densification" of H_1 sub-matrix of base code (i.e, the source sub-matrix) and maybe the decrease of the smallest stopping sets size[2]. This impact leading to the uselessness of extra-repair symbols which are used to cope the problem of the small stopping sets in the base code graph. Even with the increase of E (i.e, reduce of the GLDPC code rate), the problem is not resolved. Therefore, the increase of N1 leads to the deterioration of the ability of correction IT/RS decoding. Table II shows a different behavior for pure ML decoding[3] as the one observed for IT/RS decoding. The average decoding inefficiency ratio decreases by increasing N1 for all the GLDPC code rates. With ML decoding, the most significant performance gains are obtained by switching from $N1 = 3$ to 5 and since this value the performance improves slightly (no significant impact of N1). For a given value of N1, the average decoding inefficiency ratio decreases by increasing E per check nodes (i.e, adding extra-repair symbols nodes to the base code graph) due to the role of extra-repair symbols. This value 5 also limits the performance degradation of IT/RS decoding compared to values $N1 > 5$. Therefore in the remaining of this work we only focus on the case where $N1 = 5$.

r_G	N1	IT/RS decoding	ML decoding
$\frac{2}{3}$ (E=0)	3	1.06669	1.04225
	5	1.09682	1.00636
	6	1.12217	1.00396
	7	1.14624	1.00250
$\frac{1}{2}$ (E=1)	3	1.10487	1.01627
	5	1.22160	1.00097
	6	1.27825	1.00040
	7	1.32857	1.00009
$\frac{1}{3}$ (E=3)	3	1.17963	1.00668
	5	1.42080	1.00019
	6	1.53045	1.00007
	7	1.62660	1.00001

TABLE II
PERFORMANCE AS A FUNCTION OF N1 AND THE DECODING SCHEME (IT/RS VERSUS ML), FOR $K = 1000$ AND $r_L = \frac{2}{3}$.

4) Performance of IT/RS versus ML decoding: Now that we chose $N1 = 5$ as a value that provides a good balance, we can quantify the erasure recovery performance gains made possible by the use of ML decoding[3] as the second step of our hybrid decoding scheme.

[2]The residual set of erasures after belief propagation decoding is exactly equal to the maximum size stopping set that is a subset of the originally erased code symbols. The performance of LDPC codes is limited by the size of smallest stopping sets, i.e design of a good LDPC code graph assumes maximizing the size of their minimal stopping sets

[3]Here we refer to the ML as Maximum Likelihood decoding but not the hybrid decoding

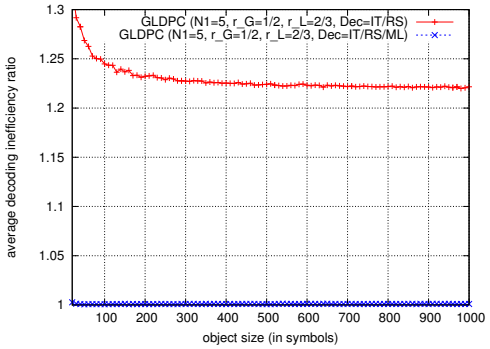


Fig. 4. Performance for (IT/RS) versus ML decoding schemes, with r_G=$\frac{1}{2}$.

Figure 4 shows the average decoding inefficiency ratio (i.e, average of 1000 GLDPC codes with $r_G = \frac{1}{2}$) versus various object sizes. This figure confirms the major performance gains made possible by the use of ML decoding[3] with GLDPC-Staircase codes for all object sizes. For instance for $K = 1000$ we obtain a gain equal to 22%. We can remark also based on Table II that for given N1 value, adding extra-repair symbols to the base graph causes the degradation of IT/RS decoding performances unlike the ML decoding. Therefore in the remaining of this work we only focus on ML decoding.

C. Average performance results

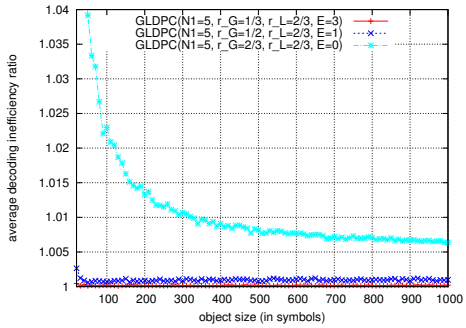
Now that the internal parameters are set, let us analyze the GLDPC-Staircase performances in various situations. We show in this section that the average performance achieved is exceptionally good, almost that of ideal codes, no matter the object size. In figure 5(a) we plot the average decoding inefficiency ratio (i.e, average of 1000 GLDPC codes), as a function of the object size, with various curves corresponding to the various code rates (i.e, r_G=$\frac{1}{3}$ ($E = 3$), r_G=$\frac{1}{2}$ ($E = 1$), and r_G=$\frac{2}{3}$ ($E = 0$), and we do the opposite in Figure 5(b).

We see in both figures that the GLDPC-Staircase codes with $E = 1$ or $E = 3$ exhibit exceptional erasure recovery capabilities, even for tiny objects. On the opposite, codes with $E = 0$ corresponding to the base codes have performance level significantly smaller for small object sizes. Hence, the addition of extra-repair symbols makes the correction capabilities of GLDPC-Staircase codes under ML decoding close to that of ideal, MDS codes, both for tiny or large objects. These tests also show that GLDPC-Staircase codes can easily achieve very small code rates while keeping exceptionally high erasure recovery performance as shown also in Table II.

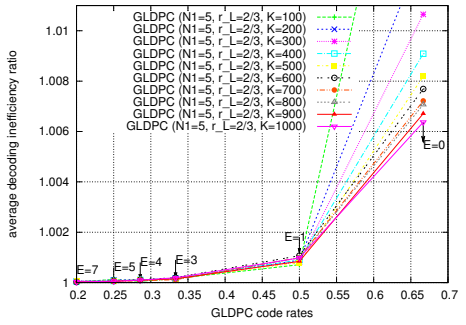
D. Decoding failure probability results

We continue the analysis with decoding failure probability, which enables us to more carefully analyze the GLDPC-Staircase codes behavior as a function of the number of received symbols and loss percentage.

Figure 6 shows the average results of 10^7 GLDPC codes with $r_G = \frac{1}{2}$ for $K = 1000$, $K = 256$, and $K = 32$. The



(a)



(b)

Fig. 5. Average decoding inefficiency ratio as a function of the object sizes (a) and code rates (b).

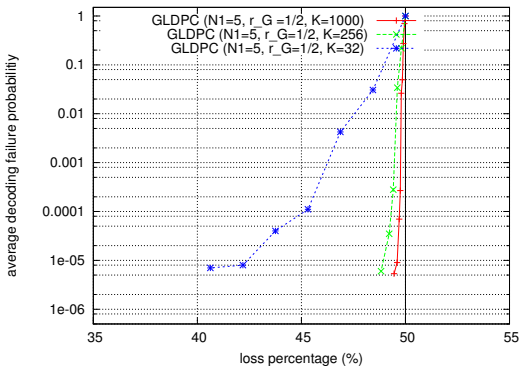


Fig. 6. Average decoding failure probability as a function of the channel loss percentage experienced for GLDPC codes ($r_G = \frac{1}{2}$) with $K = 32$, $K = 256$, and $K = 1000$.

black vertical line corresponds to ideal, MDS code, for which the decoding failure is equal to 0 as long as the experienced loss rate is strictly inferior to 50%. This figure confirms that GLDPC-Staircase codes feature a very small decoding overhead, close to that of ideal, MDS codes, with no visible error floor above 10^{-5} decoding failure. This is obvious with objects of size 1000 and $K = 256$, it remains almost true with the $K = 32$ case (i.e, error floor started from 8.10^{-6} with 42% of loss for K=32, bellow $5.33.10^{-6}$ with 49.45% of loss for K=1000).

Table III gives additional details where we performed 6.10^6, and 2.10^6 GLDPC codes ($r_G = \frac{1}{2}$) for $K = 1000$ and $K = 32$ respectively. It provides the average decoding failure probability as a function of the number of received symbols above K, i.e. as a function of the overhead. It shows that for large or small object sizes, a few symbols above K is sufficient for decoding to success with a high probability. For instance, with $K = 1000$ received symbols, 1819473 among 6.10^6 GLDPC codes can recover all the erased symbols (i.e, 69.67 % as decoding failure probability) and with two received symbols more than K, 5703206 among 6.10^6 GLDPC codes can decode all the erased symbols (i.e, 4.9 % as decoding failure probability). With $K = 32$ received symbols, GLDPC codes can decode all the erased symbols with decoding failure probability equal to 3.0584% (i.e, 1938832 among 2.10^6 GLDPC codes that can recover all the erased symbols) and with two received symbols more than K, 1999778 among 2.10^6 GLDPC codes can recover all the erased symbols (i.e, 0.011% as decoding failure probability).

K	reception overhead	average decoding failure prob.
32	0	0.0305
	1	$4.2 * 10^{-3}$
	2	$1.1 * 10^{-4}$
	3	$4 * 10^{-5}$
	4	$8 * 10^{-6}$
	5	$7 * 10^{-6}$
	6	$2 * 10^{-6}$
1000	0	0.6967
	1	0.2725
	2	0.0494
	3	0.0262
	4	$2.68 * 10^{-4}$
	5	$6.96 * 10^{-5}$
	6	$9 * 10^{-6}$

TABLE III
AVERAGE DECODING FAILURE PROBABILITY OF GLDPC-STAIRCASE CODES ($r_G = \frac{1}{2}$) UNDER ML DECODING FOR K=32 AND K=1000

IV. CONCLUSIONS

This paper has introduced the GLDPC-Staircase codes based on RS codes ("quasi" Hankel matrices) under IT/RS/ML decoding. To obtain excellent performances of these codes, we adjust firstly their internal parameters. We have shown that these codes have two key benefits: on the one hand, a significant number of extra-repair symbols can be produced on the fly, which provides "small rate" rate capabilities; on the other hand, these codes feature both a very small decoding overhead and low error floor, not only for large object, but also for very small ones. These properties make GLDPC-Staircase codes an excellent AL-FEC solution for many situations that require erasure protection. If the current paper focuses on practical performance evaluations, in [16] we introduce an asymptotic analysis, in terms of EXtrinsic Information Transfer functions and we derive an upper bound of the ML decoding threshold based on the area theorem. Both papers nicely complement with one another. Future works will focus on encoding and decoding speed and complexity evaluations, a key aspect for practical realizations.

V. ACKNOWLEDGMENTS

This work was supported by the ANR-09-VERS-019-02 grant (Adaptable, Robust, Streaming SOlutions (ARSSO)

project) and by the INRIA - Alcatel Lucent Bell Labs joint laboratory.

REFERENCES

[1] R. Gallager, "Low Density Parity Check Codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21– 28, 1962.

[2] D. MacKay and R. Neal, "Near Shannon limit performance of low density parity check codes," *IET Electronics Letters*, vol. 33, no. 6, pp. 457–458, 1997.

[3] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sept. 1981.

[4] J. Boutros, O. Pothier, and G. Zémor, "Generalized Low Density (Tanner) Codes," *IEEE International Conference on Communications (ICC'99)*, vol. 1, pp. 441–445, 1999.

[5] M. Lentmaier and K. Zigangirov, "On Generalized Low-Density Parity-Check Codes Based on Hamming Component Codes," *IEEE Transactions on Communications*, vol. 3, no. 8, pp. 248–250, 1999.

[6] G. Yue, L. Ping, and X. Wang, "Generalized Low-Density Parity-Check Codes Based on Hadamard Constraints," *IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 1058–1079, 2007.

[7] J. Chen and R. Tanner, "A Hybrid Coding Scheme for the Gilbert Elliott Channel," *IEEE Transactions on Communications*, vol. 54, no. 10, pp. 1787–1796, Oct. 2006.

[8] N. Miladinovic and M. Fossorier, "Generalized LDPC Codes with Reed-Solomon and BCH Codes as Component Codes for Binary Channels," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1239–1244, 2005.

[9] I. Djordjevic, O. Milenkovic, and B. Vasic, "Generalized Low-Density Parity-Check Codes for Optical Communication Systems," *Lightwave Technology*, vol. 23, no. 5, pp. 1939–1946, 2005.

[10] Y. Wang and M. Fossorier, "Doubly Generalized LDPC Codes," *IEEE International Symposium on Information Theory*, pp. 669–673, July 2006.

[11] E. Paolini, M. Fossorier, and M. Chiani, "Analysis of Doubly-Generalized LDPC Codes with Random Component Codes for the Binary Erasure Channel," *Proceedings of Allerton Conference on Communications, Control and Computing*, 2006.

[12] M. Cunche, V. Savin, V. Roca, G. Kraidy, A. Soro, and J. Lacan, "Low-rate coding using incremental redundancy for GLDPC codes," *IEEE International Workshop on Satellite and Space Communications*, pp. 299–303, Oct. 2008.

[13] V. Roca, C. Neumann, and D. Furodet, "Low density parity check (ldpc) staircase and triangle forward error correction (fec) schemes," jun 2008, IETF Request for Comments, RFC 5170 (Standards Track/Proposed Standard).

[14] F. Mattoussi, V. Roca, and B. Sayadi, "Complexity comparison of the use of Vandermonde versus Hankel matrices to build systematic MDS Reed Solomon codes," *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2012.

[15] R. Roth and S. Gadiel, "On Generator Matrices of MDS Codes," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 826–830, 1985.

[16] F. Mattoussi, V. Roca, and B. Sayadi, "Optimization with EXIT functions of GLDPC-Staircase codes for the BEC," *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2012.